Thermocouple logger

Plugging into a pc's printer port, Pei An's thermocouple-based temperature logger gives you six channels each capable of measurements in the range -270°C to 1370°C. Measurement resolution is 19 bits.

This article describes a low-cost six-channel temperature data logger using thermocouples. The device connects to the printer port of a pc via a standard printer cable. Six K-type thermocouples can be connected to the device using industry standard thermocouple connectors.

With the right thermocouples, temperatures in the range -270°C to 1370°C can be measured. **Figure 1** shows the complete system.

Thermocouple principles

When the junction of two dissimilar metals is heated, an e.m.f. is generated. This is known as the *Seebeck* effect and the junction is called a thermocouple.

Junctions are formed by twisting the ends of two wires together and then welding them. The basic operation of a thermocouple is shown in **Fig. 2**. The sensing junction – i.e. the hot junction – is at the temperature to be measured. The reference junction, which is the cold junction, is held at a reference temperature.

Resulting emf is proportional to the difference between the temperatures of the two junctions. The amplitude of the emf depends on the composition of the two wires.

There is a family of industrial standard thermocouples and they are identified by types, Fig. 2. Operating temperature range, composition and accuracy of thermocouples are defined in the IEC584 standard, which is called a code for temperature measurement using thermocouples.

Type K thermocouple is probably the most widely used. It is suitable for temperatures ranging from -270°C to +1370°C. Its positive arm is 95% nickel balanced by Al, Si and Mn – an alloy known as Alumel. The negative arm is an alloy known as chromel which comprises 90% nickel and 10% chrome.

Type K thermocouples have three classes of accuracy:

Class 1:	range:	-40°C to +1000°C
	accuracy:	±0.004T or ±1.5°C
Class 2:	range:	-40°C to +1200°C
	accuracy:	±0.0075T or ±2.5°C
Class 3:	range:	-200°C to +100°C
	accuracy:	±0.015T or ±2.5°C

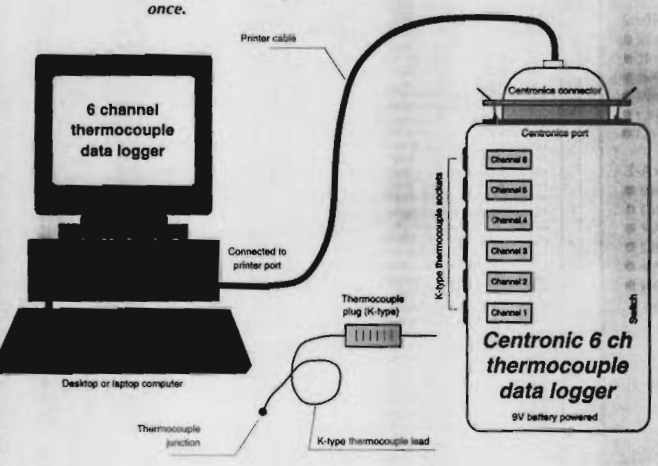


Fig. 1. Linking to the pc via its LPT port, the data logger takes readings of up to six thermocouples at once.

Since the logger is based on cmos devices, it can be implemented compactly and operated from a 9V battery.

The larger of the two deviation values should be chosen. Value T is the temperature measured by the thermocouple.

Figure 3 illustrates the emf values as a function of temperature in degrees celsius for various types of thermocouples. The output is reasonably linear over a wide temperature range. But above a certain temperature, the emf falls off. Knowing the emf, you can find the temperature using a polynomial,

$$T = A_0 + A_1X + A_2X^1 + A_3X^3 ... + A_nX^n$$

in which T is the temperature in degrees celsius, X is the thermocouple output voltage in volts, A_{0-n} are polynomial coefficients which are unique to each type of thermocouple and n is the order of the polynomial. The relationship between the e.m.f. versus temperature can be found in the IEC584 international thermocouple reference tables.

Thermocouple wires can be as small as 50μm in diameter – or even less. As a result, thermocouple junctions can be made very small, especially if a butt joint is used. Due to the low mass involved, small thermocouples can have fast response times of a few milliseconds. They are also cheap and have a wide operating temperature range.

Industrial thermocouples come in different forms. They have to be chosen to suit the individual application.

A classical way to measure temperature using thermocouples is shown in Fig. 4a. The sensing junction is at the temperature to be measured. The reference junction is placed at a reference temperature.

One way of providing the reference temperature is to use an insulated bath containing water and ice. This will give a temperature exactly at 0°C. The emf is proportional to the temperature difference between the junctions and is measured by a voltmeter.

The emf from multiple thermocouples can be measured using methods shown in Figures 4b and 4c. Such methods are especially useful for multi-channel thermocouple measurements.

These methods require the temperature of the connection terminals to be known. This can be done using a reference thermocouple connected to one of terminals, as in Fig. 4b), or using a temperature sensor attached to the terminals for mea-

Continued on page 31 after listing

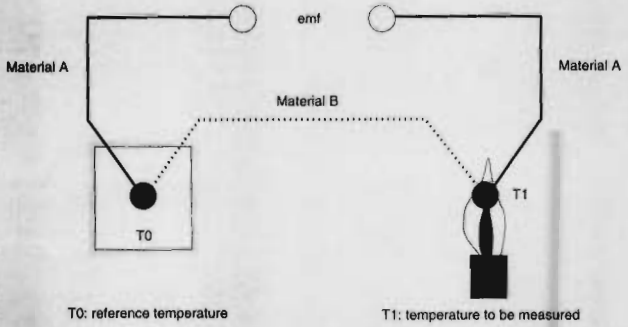


Fig. 2. In a conventional thermocouple sensing arrangement, emf produced is proportional to the difference between two thermocouple junctions – one at a reference temperature, the other at the temperature to be determined.

Thermocouple types	Materials
E	chromel vs constantan
J	iron vs constantan
K	chromel vs alumel
R	platinum vs platinum+13% rhodium
S	platinum vs platinum+10% rhodium
T	copper vs constantan

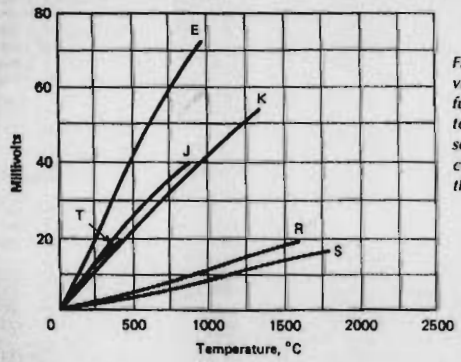


Fig. 3. EMF variations as a function of temperature for a selection of commonly used thermocouples.

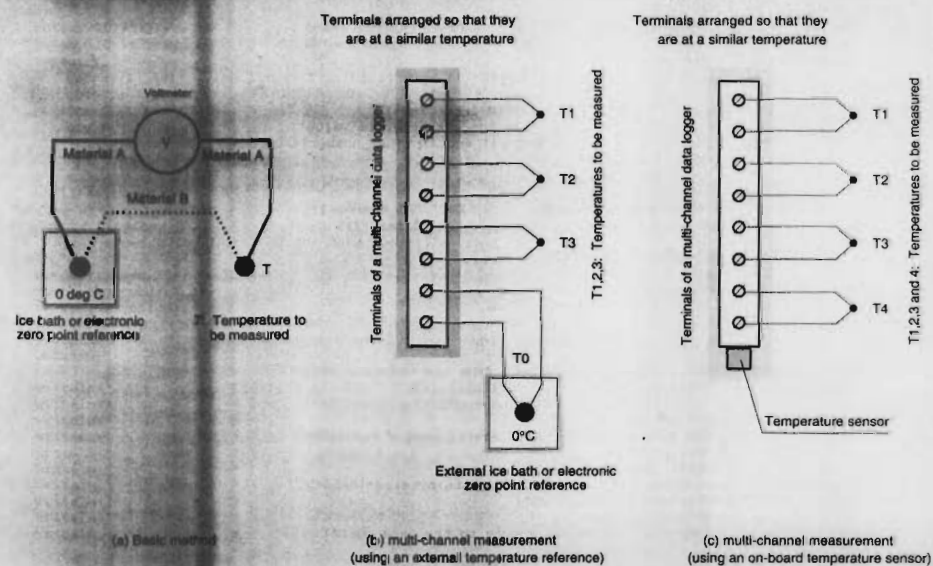


Fig. 4. In practice, there are several different ways of configuring the thermocouple circuit.

```
Program thermocouple_logger;
(*6 channel temperature logger using thermocouples
   Hardware and software developed by Dr Pei An, 1/98*)
{DATA PORT:  DB0-CLK,  DB1-A0, DB2-CONV, DB3-CAL, DB4-ADD0;
             DB5-ADD1, DB6-ADD2 DB7-ADD3
             ADD1 to ADD3 connected to A0,A2 and A3 on multiplexer
 STATUS PORT DB6-READY, DB7-DATA IN }
{  A0=0 selects expanded analogue input channel
   A0=1 selects on-board analogue multiplexer }
{  address=0 for batter voltage monitoring
   address=1 for on-board temperature sensor, 10mV/deg C
   address=2-7 for voltages inputs (6 off) }
uses
   Crt, dos;
Var
   ch,command,i:byte;
   Datax:array[1..30] of byte;
   unitx:char;
   P_address,dummy:integer;
Function bitweight(bit:byte):real;
var
   p1,i:longint;
begin
   P1:=1;
   if bit=1 then bitweight:=1
        else begin
             for i:=1 to bit-1 do P1:=P1*2;
             bitweight:=p1;
             end;
end;
(*---Detect LPT base addresses---*)
Procedure Centronics_address;
(* $000:$0408 holds the printer base address for LPT1
   $000:$040A holds the printer base address for LPT2
   $000:$040C holds the printer base address for LPT3
   $000:$040e holds the printer base address for LPT4
   $000:$0411 number of parallel interfaces in binary format*)
var
   lpt:array[1..4] of integer;
   number_of_lpt,LPT_number,code:integer;
   kbchar:char;
begin
   clrscr;
   LPT_number:=1;                    (*to set default printer*)
   number_of_lpt:=mem[$0000:$0411];
                                     (*to read number of installed Centronics ports*)
   number_of_lpt:=(number_of_lpt and (128+64)) shr 6;       (*Bit manipulation*)
   lpt[1]:=memw[$0000:$0408];        (*Memory read procedure*)
   lpt[2]:=memw[$0000:$040A];
   lpt[3]:=memw[$0000:$040C];
   lpt[4]:=memw[$0000:$040E];
   textbackground(blue); clrscr;
   textcolor(yellow); textbackground(red); window(10,22,70,24); clrscr;
   writeln('Number of LPT installed        :        ',number_of_lpt:2);
   writeln('Addresses for LPT1 to LPT 4:   ',lpt[1]:3,'     ', lpt[2]:3,'     ', lpt[3]:3,'     ',
lpt[4]:3);
   write('Select LPT to be used (1,2,3,4)  :        ');
   delay(1000);
   if number_of_lpt>1 then begin
                        (select LPT1 through LPT4 if more than 1 LPT installed)
        repeat
        kbchar:=readkey;                  (*read input key*)
        val(kbchar, LPT_number, code);    (*change character to value*)
        until (LPT_number>=1) and (LPT_number<=4) and (lpt[LPT_number]<>0);
                    end;
   clrscr;
   P_address:=lpt[LPT_number];
   writeln('Your selected printer interface:        LPT',LPT_number:1);
   write('LPT    Address                    :        ',P_address:3);
   delay(1000);
   textbackground(black); window(1,1,80,25); clrscr;
end;
(*---read data from STATUS port of pc---*)
Function Read_status_port(P_address:integer):byte;
var
   byte1:byte;
 begin
   byte1:=port[P_address+1];         (*read a byte from the status port*)
   byte1:=byte1 and (120+128);
                     (*11111000 (MSB to LSB) and 0dddd... = 0ddddd000*)
   Read_status_port:=byte1 shr 3;
                     (*shift 3 bit right, Read_status_port = 0000hhhh*)
end;
(*---Write data to DATA port of pc---*)
Procedure Write_data_port(P_address:integer; port_data:byte);
(*no lines in the Data port are not inverted*)
begin
   port[P_address]:=port_data;       (*output a byte to the data port*)
```

```
end;
Function Input_data:byte;
begin
   input_data:=read_status_port(P_address);
 end;
Function CAD:longint;
{Calibrating the A/D converter}
begin
   command:=0;
   port[P_address]:=0+8+command; {CONV=0, CAL=1}
   delay(20);
   Port[P_address]:=4+8+Command; {CONV=1 and CAL=1, calibrating started}
   repeat until ((input_data and 8)=       0) or keypressed;
                  {-REDY goes low to indicate a complete conversion}
   port[P_address]:=command;        {CONV=0, CAL=0}
   delay(2000);
end;
Function voltage(address:byte):real;
{read voltage }
var
   sum:real;
   v:array [1..10] of real;
   i,ix:integer;
begin
   sum:=0;
   command:=A0*2+address*32;
   Port[P_address]:=4+Command; {CONV=1, CAL=0}
   repeat until (input_data and 8)=       0;
                  {-REDY goes low to indicate a complete conversion}
   delay(1);
   for i:=1 to 20 do
        begin
            datax[21-i]:=1-round((input_data and 16)/16);
                  {note: DB7 of the status port is inverted}
            port[P_address]:=1+command;
                  {CLK goes from low to high to start shifting}
            port[P_address]:=command;
                  {CLK goes from high to low to clock out next bit}
        end;
   for i:=1 to 20 do begin sum:=sum+bitweight(i)*datax[i] end;
   voltage:=(sum - 8*256*256) /8/256/256*2.500; {bipolar shifting}
   port[P_address]:=command;
   delay(100);
end;
Function TC(channel:byte):real;
(*read temperature from a channel, channel=1 to 6*)
var
   temp:real;
begin
   temp:=(voltage(1,1)/0.01 + (voltage(1,channel+1))/0.0000405);
   TC:=temp;
   if (temp>=1500) then TC:=0;
   if (temp<=-100) then TC:=0;
end;
Procedure Measure_all;
begin
   gotoxy(10,25); write('          Press any key to stop logging');
   gotoxy(10,1) ; write('          Data read from the data logger');
   repeat
      gotoxy(21,10); write('Battery voltage     :',voltage(1,0)*5.7:10:1,'  [V]');
      gotoxy(21,11); write('On-board temp.:',voltage(1,1)/0.01:10:1,'       [deg C]');
      for i:=1 to 6 do
      begin
            gotoxy(21,11+i);
            write('Temperature ',i,'      :',TC(i):10:1,'  [deg C]');
      end;
   until keypressed;
end;
Procedure Diagram;
{A diagram showing the layout of the data logger}
begin
   window(1,1,80,25);
   Textbackground(blue);
   textcolor(yellow);
   clrscr;
   writeln('       Layout of the 6 channel thermocouple data logger');
   writeln;

   writeln('             ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ»                     Features');
   writeln('             º Computer Thermocouple Logger     º    ********************************');
   writeln('             º                                  º    ß      6 K-type thermocouple sockets');
   writeln('             º                                  º    ß      19-bit A/D conversion accuracy');
   writeln('      <<<<   º          Channel 6               º    ß      Measurement range');
   writeln('             º                                  º           -100 to 1350°C ');
   writeln('      <<<<   º          Channel 5               º           ±2°C accuracy');
   writeln('             º                                  º    ß      Centronics port connection');
   writeln('      <<<<   º          Channel 4               º    ß      On-board temperature sensor');
   writeln('             º                                  º    ß      A PP3 9V battery required');
```

```
   writeln('      ====    º       Channel 3                    º      ß     Battery low voltage warning');
   writeln('              º                                    º      ');
   writeln('      ====    º       Channel 2                    º      ß     Slim size');
   writeln('              º                                    º      ');
   writeln('      ====    º       Channel 1                    º      ');
   writeln('              º                                    º            Applications');
   writeln('              º                                    º      ');
   writeln('              º                                    º      ß     Temperature monitoring');
   writeln('              ¶                                    º');
   writeln('              º                    .               º      ');
   writeln('              Èííííííííííííííííííííííííííííííí_           ');
   writeln('                                                          ');
   writeln('                                                   ');
   write ('                                                    Press any key to continue');
end;
Procedure manual;
begin
   window(1,1,80,25);
   textbackground(blue);
   textcolor(yellow);
   clrscr;
   gotoxy(10,1) ; write('          6 channel thermocouple data logger');
   gotoxy(10,2) ; write('          Chose the number and pressed RETURN');
   gotoxy(20,10); write('[1]       Diagram of the data logger');
   gotoxy(20,11); write('[2]       Calibration');
   gotoxy(20,12); write('[3]       Measure all');
   gotoxy(20,13); write('[4]       Quit the session');
   textbackground(green);
   textcolor(white);
   window(1,25,80,25);
   clrscr;
   gotoxy(1,10); write(' Press [1] to [4]');
   gotoxy(80,25);
end;
Procedure manual_selection;
var
   key_char:char;
begin
   manual;
repeat
   key_char:=readkey;
   if key_char='1' then begin
                     diagram;
                     repeat until keypressed;
                     readln;
                     manual;
                     end;
   if key_char='2' then begin
                     window(1,1,80,25);
                     dummy:=CAD;
                     manual;
                     end;
   if key_char='3' then begin
                     window(1,1,80,25);
                     textbackground(blue);
                     textcolor(yellow);
                     clrscr;
                     measure_all;
                     manual;
                     end;
   if key_char='4' then begin
                     halt;
                     end;
until (key_char)='7';
   delay(1000);
   clrscr;
end;
(*****MAIN PROGRAM*****)
begin
   Centronics_address; (*select a Centronics interface*)
   Port[P_address]:=0; {all lines of the DATA port are zero}
   command:=0;
   clrscr;
   dummy:=cad;
   delay(10000);
   dummy:=cad;
   delay(10000);
   dummy:=round(voltage(1,0));
   clrscr;
   manual;
   manual_selection;
end.
```

Software driver

The software driver is written in Turbo Pascal 6 List. 1. This program only demonstrates how the hardware of the device is controlled by a software.

It shows how to calibrate the a-to-d converter and how to get a conversion data from the converter. It does not have functions to log data continuously and save the data into a file.

Here's an outline of what the functions and procedures do. The procedure,

`Centronics_address`

detects the number of Centronics ports installed on your computer and allows you to select which port to use. The function,

`Read_status_port(P_address:integer)`

reads data from the status port. The procedure,

`Write_data_port(P_address:integer; port_data:byte)`

outputs a data byte from the data port. The function,

`CAD:longint`

performs calibration of the a-to-d converter. The function,

`voltage(address:byte)`

controls the a-to-d converter and measures a voltage from an analogue channel specified by address. The function,

`TC(channel:byte)`

converts the voltage into a temperature value. Here the relationship between e.m.f. and temperature is treated as a linear one. Finally, the procedure,

`Measure_all`

measures temperatures from all six channels.

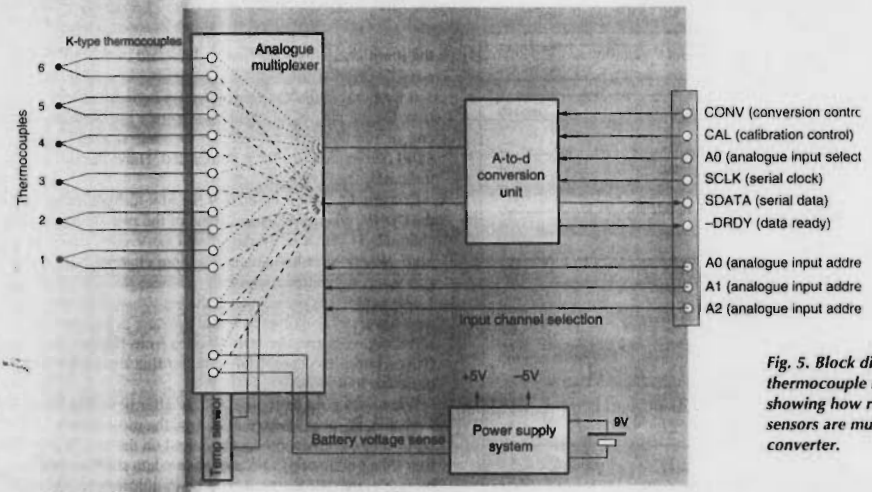


Fig. 5. Block diagram of the thermocouple measurement subsystem showing how readings from the six sensors are multiplexed into one a-to-d converter.

suring the temperature, Fig. 4c.

All the terminals should be kept at a similar temperature to minimise errors in the reference temperature. This can be done by mounting all the terminals on a piece of copper plate and keeping them away from heat sources.

Overview of the hardware

Figure 5 is a block diagram of the thermocouple data logger. The device consists of four units: the a-to-d conversion unit, the analogue multiplexer unit, the temperature sensing unit and the power supply unit.

The a-to-d conversion unit is based on a *CS5504* 20-bit a-to-d converter. When it operates in bipolar mode and the voltage reference is 2.5V, it could measure an input voltage as small as ±5.45μV. Note that the K-type thermocouple gives 40μV/°C.

In order to measure six channels of thermocouple signals, the data logger uses a *MAX337* analogue multiplexer. This device provides eight differential analogue inputs. Six of them are used for thermocouples. One is used for the on-board temperature sensor and one is used for monitoring the voltage of the battery.

The temperature sensing unit is based on an *LM35DZ* celsius temperature sensor. It measures the temperature of the terminals on the circuit board. This is the reference temperature for calculating temperatures measured by thermocouples.

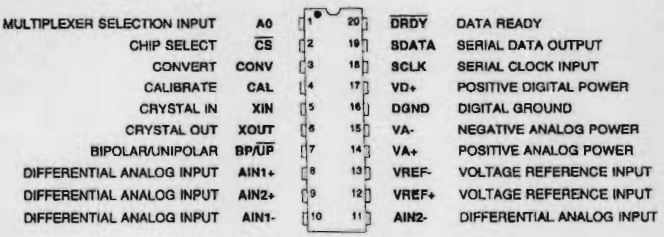
The power supply unit comprises a low-power, low-voltage-drop +5V *HT1050* voltage regulator and a *7660* voltage inverter to generate a –5V supply from the +5V power supply. The –5V power supply is used by the *CS5504* and the *MAX337*.

The CS5504 a-to-d converter

For a-to-d conversion, I chose the dual-channel 20-bit *CS5504*. This device uses delta-sigma conversion and has serial i/o. It provides low-cost, high-resolution measurement at output word rates up to 200 samples per second.

Being a cmos device, the *CS5504* draws less than a milliamp from the supply. It consists of a delta-sigma converter, a voltage reference, a calibration microcontroller, a static ram, a digital filter and a serial interface, **Fig. 6**.

The on-chip digital filter provides mains rejection at 50Hz and 60Hz when the device is operated from a 32.768kHz crystal, which gives a 20Hz sampling rate. The on-chip self-calibration circuitry ensures minimum offset and full-scale errors in a conversion.



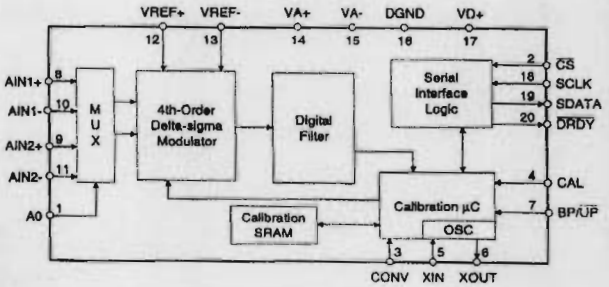


Fig. 6. At relatively low cost, the CS5504 cmos a-to-d converter provides 20-bit measurement resolution and has on-chip mains rejection filtering.

Three operating states are possible with the device: stand-by, calibration and conversion. Stand-by state is entered after the device has completed an operation and no command is given to it. After a power-on, a wake-up period comprising 1800 clock periods exists before the device enters the stand-by state.

Calibration must be performed before a valid conversion can be made. The calibration state is entered when CAL is high and CONV goes from low to high. During calibration, the device first performs an offset calibration and then a gain calibration. This is conducted by the on-board calibration micro-controller. Calibration takes 3246 clock cycles and the calibration coefficients are stored in the calibration static ram for use during conversion.

At the end of the calibration cycle, the microcontroller checks the logic state of the CONV signal. If it is low, the device enters the stand-by mode, waiting for further instruction. If it is high, the device performs conversion on one of

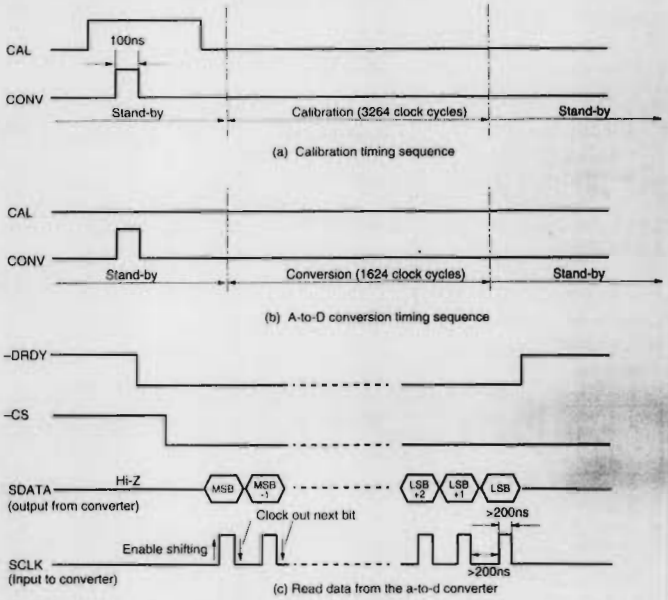


Fig. 7. Timing sequences for the CS5504 20-bit digitiser chip.

the input channels specified by the state of A_0 at the rising edge of CONV signal.

If CAL and CONV signals are both high, after the calibration, the device automatically performs a conversion. At the end of the conversion, –DRDY become low to indicate a valid conversion has been completed. Here, the minus sign indicates negative logic.

Conversion state is entered at the low-to-high transition of the CONV, when CAL is low. One of the two analogue input channels is selected by A_0 when CONV goes from low to high. When A_0 is low, analogue input channel 1 is selected and when A_0 is high, channel 2 is selected.

The CONV line is kept high during the conversion. After the conversion is completed and the data is latched into the serial data register, the –DRDY goes from high to low. This flags a completed conversion. After that the CONV can be pulled to logic low.

When –CS goes from high to low after new data becomes available, i.e. after –DRDY goes low, the most-significant bit, DB_{19}, of the conversion data is output on the SDATA pin. The first rising edge of SCLK enables the data shifting. In stand-by mode, the SCLK line should be held low.

At the falling edge of the SCLK, the next data bit appears on the SDATA pin. Each subsequent falling edge shifts out the serial data bits. Once the least-significant bit of the conversion data is preset, the falling edge of SCLK causes the SDATA output to go to tri-state and –DRDY to return high

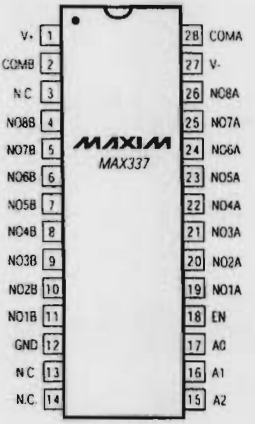


Fig. 8. MAX337 is a dual eight-into-one multiplexer featuring an on resistance of 400Ω and less than 20pA leakage current at room temperature.

MAX337

A2	A1	A0	EN	ON SWITCH
X	X	X	0	NONE
0	0	0	1	1
0	0	1	1	2
0	1	0	1	3
0	1	1	1	4
1	0	0	1	5
1	0	1	1	6
1	1	0	1	7
1	1	1	1	8

LOGIC "0" = $V_{AL} \leq 0.8V$, LOGIC "1" = $V_{AH} \geq 2$

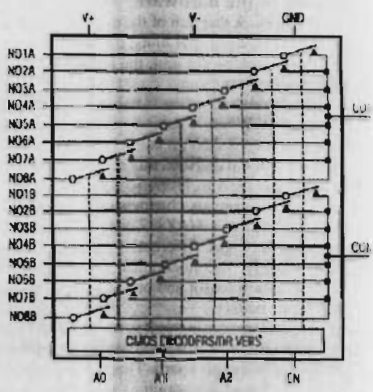
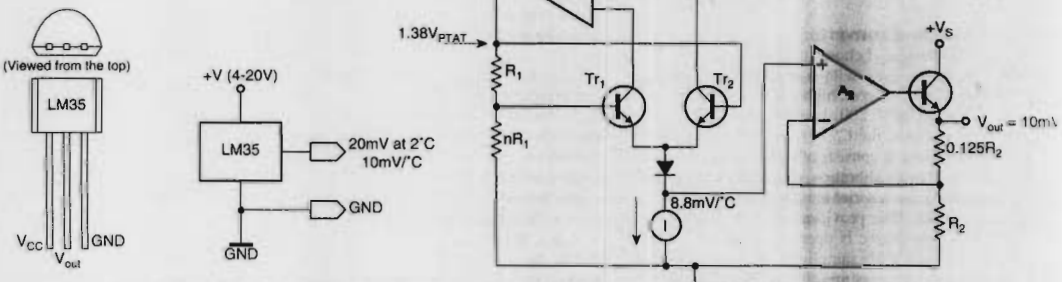
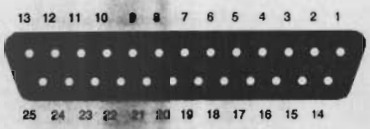


Fig. 9. Details of the temperature sensor used for the reference temperature.



(a) Pin-out of LM35DZ (b) Temperature sensor based on LM35DZ (c) Internal block diagram



(a) Pin-out of the Centronics connector on pc compatibles - viewed from the back of the pc
Connector type: 25-pin female D-type

(b) Pin-out of the Centronics connector on printers - viewed from the back of the printer
Connector type: 36-pin female Centronics-type

Fig. 10. Functions of a standard pc's LPT port connections follow the long-established Centronics standard.

The Centronics printer port

Pin layout and functions of a pc's Centronics printer port is shown in **Fig. 10**.

Details of the port can be found in reference 3. In brief, a standard Centronics port contains three i/o ports. One port is the DATA port, comprising eight outputs, another is the CONTROL port which consists of four outputs and the third is the STATUS port which is made up of five inputs.

The DATA and CONTROL ports are output ports and the status port is an input. The computer uses these ports to output data and the STATUS port to input a five-bit word.

Pin functions of the Centronics port connectors

Connector		Direction	Name	Explanation
computer	printer			
1	1	C to P	$\overline{\text{STROBE}}$	Strobe data
2	2	C to P	DB0	Data bit 0
3	3	C to P	DB1	Data bit 1
4	4	C to P	DB2	Data bit 2
5	5	C to P	DB3	Data bit 3
6	6	C to P	DB4	Data bit 4
7	7	C to P	DB5	Data bit 5
8	8	C to P	DB6	Data bit 6
9	9	C to P	DB7	Data bit 7
10	10	P to C	$\overline{\text{ACK}}$	Indicating data received
11	11	P to C	BUSY	Indicating printer busy
12	12	P to C	PE	Indicating paper empty
13	13	P to C	SLCT	Indicating printer on line
14	14	C to P	$\overline{\text{LF/CR}}$	Auto linefeed after carriage return
15	32	P to C	$\overline{\text{ERROR}}$	Indicating printer error
16	31	C to P	$\overline{\text{INITIALISE}}$	Initialise printer
17	36	C to P	$\overline{\text{SLIN}}$	Select/deselect printer
18-25	19-30 and 33		GND	Twisted-pair return ground
	18,34		Unused	
	16		Logic GND	Logic ground
	17		Chasis GND	Chasis ground

'C' = Computer 'P' = Printer

Table 1. Output coding for unipolar and bipolar modes.

Unipolar mode	Output codes	Bipolar mode
$>V_{ref}$–1.5 lsb	$FFFFF_{16}$	$>V_{ref}$–1.5 lsb
V_{ref}–1.5 lsb	$FFFFF_{16}$	V_{ref}–1.5 lsb
$V_{ref}/2$–0.5 lsb	80000_{16}-$FFFFF_{16}$	–0.5 lsb
0.5 lsb	00001_{16}-00000_{16}	$-V_{ref}$+0.5 lsb
<0.5 lsb	00000_{16}	$<V_{ref}$+0.5 lsb

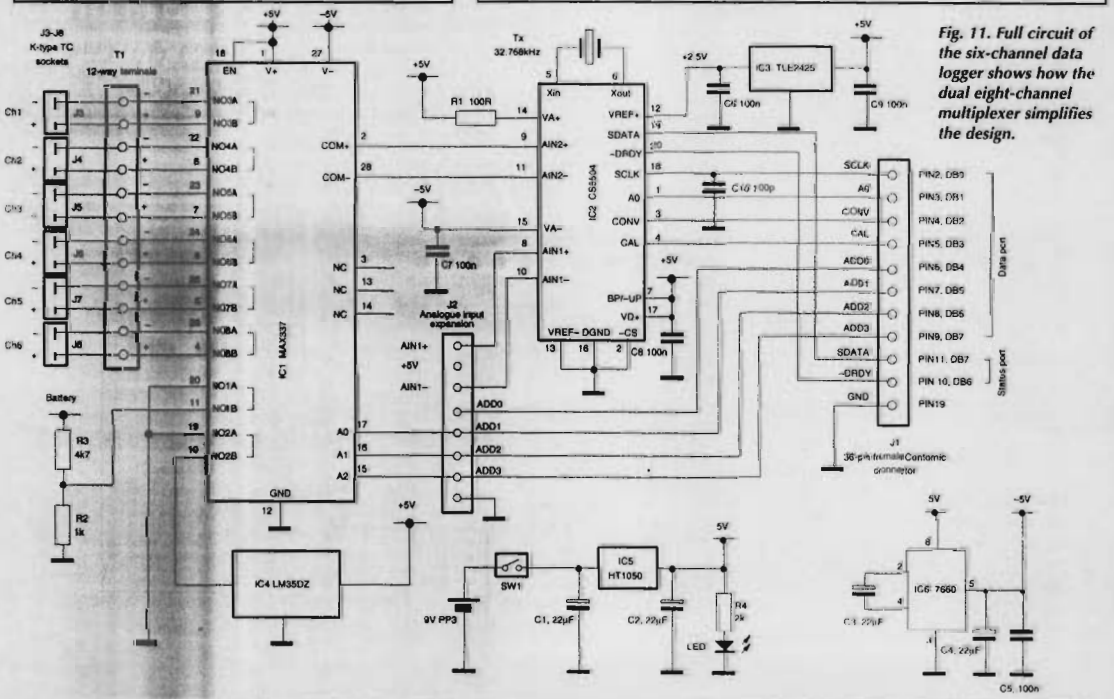


Fig. 11. Full circuit of the six-channel data logger shows how the dual eight-channel multiplexer simplifies the design.

This indicates that the serial data register is emptied. Only under this condition can the serial port registers be updated with new data on completion of another conversion, **Fig. 7**.

The input signal can be configured for a unipolar or bipolar signal depending on the status of BP/–UP line. The *CS5504* converter outputs data in a binary format when converting unipolar or in offset binary form when converting a bipolar signal. **Table 1** shows the output coding for the two measurement modes.

More details on the *CS5504* are available in the manufacturer's data sheet.[1]

Eight-into-one multiplexing

The *MAX337* is a dual eight-to-one cmos analogue multiplexer. It features a maximum on resistance of 400Ω and the device conducts current equally well in both directions. It has an extremely low off leakage – less than 20pA at room temperature – and an on channel leakage figure of less than 50pA.

The device operates from a single supply of between +4.5V and +30V, or from a dual power supply of ±4.5V to ±20V. All control pins are ttl compatible.

Pin-out, pin functions and internal block diagram of the *MAX337* are given in **Fig. 8**. The device contains two eight-to-one analogue multiplexers. One of the eight inputs is connected to the common output by control of a three-bit binary address, A_{0-2}.

In the present circuit, the two multiplexers are used together to provide differential multiplexed inputs. More details on the *MAX337* can be found in the maker's data sheet.[2]

Temperature referencing

An *LM35DZ* is used as the on-board temperature sensor, Fig. 9. This is a precision temperature sensor whose output voltage is linearly proportional to the celsius temperature in a range from 0°C to 100°C. Without calibration, it is able to achieve an accuracy of 0.6°C at 25°C.

The sensitivity of the sensor is 10mV per degree celsius. The device requires a supply from +4V to 30V. Its quiescent current is only 56µA with a voltage supply of +5V. This makes the sensor ideal for battery operated applications. Low quiescent current also ensures that the self-heating of the device is below 0.08°C in still air.

Circuit detail

You will see from **Fig. 11** that the DATA and the STATUS ports of the Centronics port are used for controlling the operations of the data logger and for reading data from it.

Lines in the DATA port are used as shown in **Table 2**. Two lines in the STATUS port are used for receiving data from the logger into the computer. Lines DB_{3-5} are not used, DB_6 connects to the –DRDY line of the a-to-d converter and DB_7 connects to the SDATA line of the a-to-d converter. This line is inverted.

Figure 12 shows how compact the system can be. ■

References

1. Data sheet CS5504, Crystal Semiconductor Corporation.
2. Data sheet MAX336/MAX337, Maxim Corporation.
3. PC Interfacing – Using Centronics, RS232 and game ports, Pei An, Newnes, Butterworth-Heinemann, 1998, ISBN0240514483.

Table 2. Summary of the functions of the printer port data pins.

DB0: serial clock SCLK for a-to-d conversion
DB1: analogue multiplexer selection line, A_0 on a-to-d converter,
DB2: conversion line CONV on a-to-d converter
DB3: calibration line CAL on a-to-d converter
DB4: ADD0, not used
DB5: ADD1, connects to *MAX337* pin A_0
DB6: ADD2, connects to *MAX337* pin A_1
DB7: ADD3, connects to *MAX337* pin A_2

Technical support

Designers' kits including all necessary components to construct a complete thermocouple data logger and the TP6 source codes are available from the author. Please make your enquiry to Dr Pei An, 11 Sandpiper Drive, Stockport, Manchester SK3 8UL, UK. Tel/fax/answer m/c +44-(0)161-477-9583. Pei's e-mail address is PAN@FS1.ENG.MAN.AC.UK.

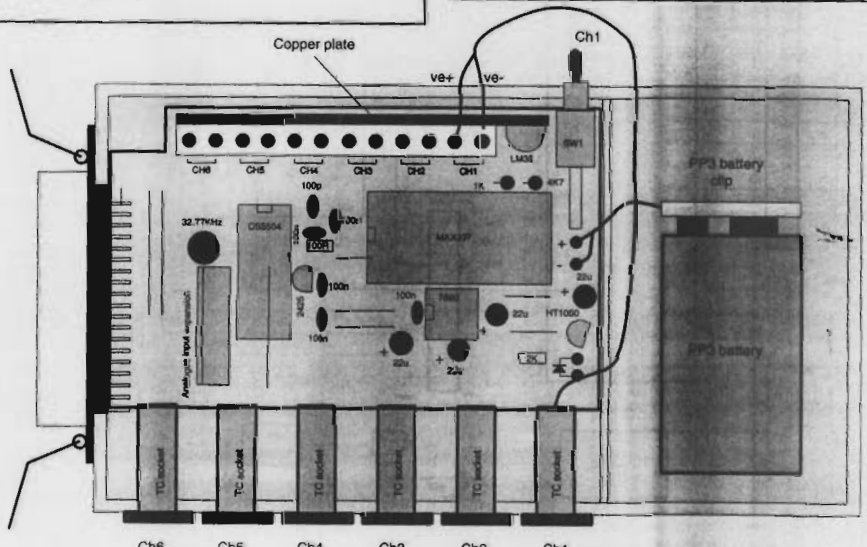


Fig. 12. Pei's self-contained thermocouple measurement subsystem is a compact design running from its own power source. As a reader service, the design is being made available from the author, together with its software.